REMARKS

Pending claims

Claims 1-32 are still pending, of which claims 1, 27, and 28 are independent claims.

Specification Amendment

The applicants request deletion of the phrase "so that the virtual OS cannot determine the presence of the VMM" from paragraph 0028 because such a definition, although accurate in describing many modern VMMs and equivalents, does not properly describe all the versions of VMMs found today, or even in the prior art at the time the present application was filed. See, for example, U.S. Patent No. 6,496,847, "System and method for virtualizing computer systems," which issued to one of the present applicants (Bugnion) on 17 December 2002 for a summary – with references -- of various known software components that act as an interface between a virtual machine and the underlying hardware. As the applicants state in paragraphs 0028 and 0030 of the present application, "The general features of VMMs are known in the art and are therefore not discussed in detail here," ... This invention works with all such configurations, the only requirement being that I/O requests by any VM, and the returned results of the requests, should be able to be tracked and intercepted by some software component that performs the functions of the VMM described below." The requested change thus does not add any new matter to this application.

Voluntary Claim Amendments

Several of the claims recited the term "virtual machine monitor" or its abbreviation "VMM." As explained above, paragraph 0030 states that the invention may be implemented using any "software component that performs the functions of the VMM described below." These claims have been amended so that they will conform to this supporting statement in the specification and to prevent needlessly limiting interpretation of the concept of a VMM.

Claim Objections

The Examiner objected to claim 16 for having a superfluous comma and to claim 27 for lacking a comma. The comma in claim 16 has been deleted and a comma has been added to claim 27 as required.

Claim Rejections Under 35 U.S.C. §103

The Examiner rejected claims 1-3, 9-11 and 25-32 under 35 U.S.C. 103 as being unpatentable over US 2002/0143842 (*Cota-Robles*) in view of US 6,584,612 (*Mueller*). Echoing claim 1, the Examiner wrote (emphasis added) that

> Cota-Robles teaches the invention as claimed, including ... performing a transformation of I/O data passing between the VM and the device (paragraphs 0015, 0027, 0047);...
>
> Mueller teaches the invention as claimed, ***including a data transformation being adjunct*** to necessary completion of an I/O request (col. 3, lines 43-45, 54-65; col. 6, lines 58-67).

As the applicants pointed out in the response to the previous Office action, such transformations as *Cota-Robles* performs on any form of data (assuming that addressing and register-identifying information is "data") are not adjunct to completion of I/O requests issued to the host processor soft device, but rather are necessary "to perform an operation requested by the guest OS" (paragraph 0025); "to complete the task that the guest OS 106 or 114 requested," (paragraph 0026); "in order to effect the operation requested by the guest OS" and "in order to complete the requested operation" (both in paragraph 0027); "to perform the read and or write operation requested by the virtual machine," "to perform an operation requested by the virtual machine," "in order to complete the task intended by the virtual machine," "according to the operation requested by the virtual machine" (all in paragraph 0029); etc. In general, *Cota-Robles* relates to a mechanism for trapping and re-directing I/O requests (see, for example, paragraphs 0038, 0039, 0042, 0044, and 0047) directed to one device (which may be implemented in software) so that the request can be carried out by another device.

The Examiner now cites *Mueller* as an example of prior art that teaches adjunct data transformation. This is not so.

First, note that *Mueller* relates to a Java virtual machine (VM), which is not the same type of "virtual machine" that the applicants' invention is directed to. As is well known, a Java VM is a substantially platform-independent environment in which a run-time engine loaded onto a given platform converts Java bytecode into the actual machine language for execution. A Java VM assumes the presence of a host operating system, for example, to which it issues calls, but it does not include an operating system of its own as a guest OS.

Even ignoring the different concepts of "virtual machine" used in *Mueller* and in the applicants' invention, *Mueller* himself explains why the address translation mechanism he provides is ***necessary*** to completion of the only type of I/O request he discusses, namely, the fetching of data stored in a device such as a read-only memory (ROM):

> Col. 2, line 51, to col. 3 line 8:
>
> Existing class loader functionality assumes the existence of a file system, and that the CLASSPATH environment variable or "java.class.path" system property will identify a location in that file system where the class loader can dynamically search for desired class files and resources. However, a proliferation of new computing devices are being developed which are intended for mobile or portable use. [Many of] these devices ... are designed without the overhead of disk drives and the file system software used to access disk files. ...
>
> Resource files may be stored in the read-only memory of these embedded devices, given the absence of disk drive storage. However, the existing Java class loader mechanism preferentially searches for stored files in a file system, as previous described. Having no file system, an alternative technique must be provided for these embedded devices. This technique must efficiently locate resources needed by an application program executing on the embedded device.

Essentially, *Mueller* teaches a form of look-up table that is used to present to a Java class loader (the ROM class loader) the appearance of an assumed file system so that the "existing [Java] application code can transparently (without modification) ***access*** resources that have been stored in ROM" (col. 3, lines 25-27, emphasis added). As just one example, see col. 7, line 56, to col. 8, line 4:

> From this information, the utility program constructs a table to be used for accessing the stored resources, where each entry in the table (or other storage mechanism, such as an array) comprises a key value set to the string identifier, and a

> value that preferably comprises (1) a pointer to the location of the resource and (2) the resource length. ... What is required in this situation is that the identifiers, locations, and length values for the stored resources are known. A table can then be constructed using this information.

Mueller specifically does *not* want any form of adjunct modification to the data (a "resource") that is accessed in ROM since this would defeat his stated goal of maintaining transparency to the calling application. See, for example, col. 10, line 65, to col. 11, line 5:

> Thus, it can be seen that the present invention provides an advantageous technique for loading resources from ROM storage in a manner that is transparent to an application program. This avoids the need for modifying existing application code when storing resources in ROM, such that the application continues to execute, unaware that the resources are in storage other than "conventional" storage such as a file system or a Web server.

As further evidence, consider *Mueller's* own example, found in column 8, in which an application wants to access the "resource" named "*picture.gif*," that is, an image file. As *Mueller* explains, when the application issues an I/O request for *picture.gif* using a getClass.getResourceAsStream("*picture.gif*") call, *Mueller's* system converts the *call* -- not *picture.gif*! – into one that invokes his loader RomLoadingClassLoader, thus:

RomLoadingClassLoader.getClass.getResourceAsStream("picture.gif")

After constructing an object identifying resource location and length, *Mueller's* system then streams *picture.gif* to the requesting application such that "***the application reads the data just as if it had come from a file stored in a file system***" (col. 8, lines 41-42, emphasis added). Note in particular that *Mueller* does not indicate any alteration of the contents of *picture.gif* in any way. Observe also that the result of *Mueller's* "transformation" would by design *not* alter the appearance of *picture.gif* on a user's display in any way. That the user is unaware that *Mueller's* class loader has been invoked (col. 3, lines 43-45) is irrelevant to the distinction between *Mueller* and the applicants' invention: *Mueller's* call transformation (a change of address or locator information as opposed to I/O content transformation) is still necessary to complete any I/O request directed to the ROM -- *Mueller's* transformation is *not* adjunct to the completion of the request, but rather the whole assumption underlying *Mueller* is that the transformation is a necessary part of such completion.

In contrast, and as explained in the response to the previous Office action, the transformations performed by the invention are separate from any operation that is required to complete the I/O request as such, that is, as issued by the requesting VM. In fact, as is mentioned in the specification (see, as one of many example, paragraph 0054), not only are the transformations according to the invention not part of any request as such made by the VM, but they may also not even be something the VM user wants. In short, both *Cota-Robles* and *Mueller* return to the requesting entity exactly the I/O data that is requested, albeit from a location, or type of device, either unlike that assumed or lacking an assumed structure (such as a file system). The applicants' invention, however, transforms (for example by partial or total deletion, addition or alteration) the I/O data *returned* or *passed on* to the requesting entity, or to some other entity (such as a video card for a display).

Both *Cota-Robles* and *Mueller* therefore lack the feature, recited in claims 1 and 28, of the I/O data "transformation being adjunct to necessary completion of the request, as issued, for the I/O operation". A similar limitation is found in claim 27, which recites that the "replacement data [is] entered as a processing step that is adjunct to the necessary completion of the I/O operation." This key feature of the invention is therefore not only not taught by *Cota-Robles* or *Mueller*, but is in fact the ***opposite*** of what these prior art references teach. As such, claims 1, 27 and 28 should be allowable over *Cota-Robles* and *Mueller* in that they recite a feature that is not found in either reference, and that provides a different outcome of an I/O request, with a unique advantage.

In paragraph 7 of the Office action, the Examiner wrote:

> Cota-Robles discusses application of data transformation to every I/O request issued by the virtual machine. This step allows the soft device to interact with hardware of the physical machine. *Mueller* discusses additional features of virtual machines that need improvement, i. e. a fast way of loading resources in the form of images, bitmaps, resources embeds specific data in a user ***display*** in a manner that is hidden from the user. Accordingly, it would have been obvious to one of ordinary skill in the art to combine *Cota-Robles* and *Mueller* since a specialized class loader would provide

> many benefits, such as allowing client to quickly load resources without interfering with other applications. Alternatively, it could be implemented in a corporate intranet, such that a system administrator can easily control the satellite workstation without risk of the user circumventing the deployment of data.

The applicants respectfully observe that this assertion is inaccurate or at best inapplicable. First, as explained above, *Mueller* never embeds anything in the requested resource (file) itself, although he changes (by pre-pending a call to his class loader) the Java call used to fetch it. Thus, for example, a user will see *picture.gif* on his display unaltered. Second, speed of resource loading (such as inputting an image file) is not an issue addressed by the applicants' invention as claimed or, in fact, by *Cota-Robles* or *Mueller.*

Third, in both *Cota-Robles* and *Mueller*, there is never an issue of a user circumventing "deployment of data." In *Cota-Robles*, the user is allowed to access all resources he normally would, the only difference being the nature of the processor used to carry out certain operations, namely, a "host processor soft device that [is] independent of the host operating system" (*Cota-Robles*, paragraph 0004). In *Mueller*, the user would not even want to circumvent the special Java class loader, because doing so would make it impossible for data to be retrieved from ROM.

As support for his assertion that it would be obvious to combine the teachings of *Cota-Robles* and *Mueller*, the Examiner is therefore hypothesizing scenarios and benefits that are tangential to the purpose of both references, that are never mentioned in either, and that in fact may not even be practicable: A system administrator would not generally control the running of a satellite workstation's process at the level of choosing whether or not to invoke *Mueller's* class loader, since the very assumption underlying *Mueller* (a storage device without a file system) is that his class loader or some similar address-translation mechanism is required for the Java VM to retrieve data at all.

The Examiner rejected the remaining claims as being obvious in view of hypothetical combinations of *Cota-Robles*, *Mueller*, and various other secondary references. By definition, these claims, all of which were dependent on claim 1, 27 or

28, include the limitations of their respective base claims. As such, *all* of the pending claims now either explicitly recite or incorporate the limitation that the transformation performed in response to the requested I/O operation is adjunct to the necessary completion of the operation. As explained above, both *Cota-Robles* and *Mueller* fail to teach this, and in fact teach the opposite. None of the cited secondary references teach this feature either. Consequently, no combination of *Cota-Robles*, *Mueller* and any of the secondary references teaches the invention as defined in the independent claims. Accordingly, the applicants respectfully submit that all of the claims now distinguish the invention over the cited prior art, and should be allowable.

Conclusion

The various embodiments of the applicant's invention as defined in the corresponding independent claims recite features that are not found at all in any of the cited references, whether the references are viewed independently or in combination. As such, the independent claims should now be allowable over the cited prior art. The various dependent claims of course simply add additional limitations and should therefore be allowable along with their respective independent base claims.

Date: 12 August 2005

Respectfully submitted,

VMware, Inc.
3145 Porter Drive
Palo Alto, CA 94304
Phone & fax: 360-793-6687

Jeffrey Pearce

Jeffrey Pearce
Reg. No. 34,729
Attorney for the Applicants